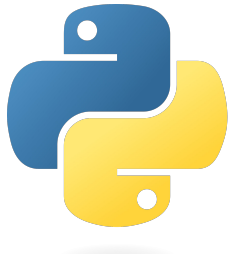

Curso de Introdução a Python
Aula 02: Tipos de dados

Ana Luiza Martins Karl

Sumário

01

Dados numéricos

- inteiros (int)
- ponto flutuante (float)
- números complexos (complex)

02

Sequências de caracteres

- strings (str)

03

Dados lógicos e especializados

- booleano (bool)
- nonetype (null)
- outros tipos

04

Coleções de dados

- listas (list)
- tuplas (tuple)
- dicionários (dict)

Tipos de dados em Python

Variáveis em programação são essenciais para armazenar e manipular dados.

Variável = nome associado a um espaço de memória

Dois tipos de dados em Python: **simples** e **compostos**

ints, float, complexos, booleanos, string, etc

listas, tuplas, dicionários, conjuntos, etc

Tipos de dados em Python

Link para notebook:
https://encurtador.com.br/cIPSX

Dados numéricos

- Inteiros (int)
- Ponto flutuante (float)
- Complexos (complex)

Dados numéricos: int

O tipo inteiro (int) é um tipo composto por caracteres numéricos inteiros – *i.e.*, podem ser escritos sem um componente decimal, podendo ter ou não sinal

Por exemplo: -22, 4, -9, 0, 10854, etc

```
1  idade = 18
2  ano = 2002
```

Dados numéricos: float

É um tipo composto por caracteres numéricos decimais, usado para números racionais (números que podem ser representados por uma fração)

Por exemplo: -0.5, 3.14, 1.0, etc

```
1  altura = 1.80
2  peso = 73.55
```

Dados numéricos: complex

É um tipo usado para representar números complexos - geralmente usados para cálculos geométricos e científicos.

O tipo complexo é composto por duas partes: a parte real e a parte imaginária

Por exemplo: -4 + 3i, 8, 16i, etc

```
1  a = 5+2j
2  b = 20+6j
```

Sequências de caracteres

- Strings

Sequências de caracteres: string

É um conjunto de caracteres dispostos numa determinada ordem, geralmente utilizada para representar palavras, frases ou textos.

Por exemplo: ‘Universidade Estadual do Norte Fluminense’, ‘Av. Alberto Lamego, 2000’

```
1  nome = 'Guilherme'
2  profissao = 'Engenheiro de Software'
```

Dados lógicos e especializados

- booleano (bool)
- nonetype (null)
- outros tipos

Dados lógicos: bool

É um tipo de dado que pode assumir somente dois valores: **True** ou **False**.

Normalmente usados em estruturas de decisão, comparações e retorno de funções.

```
1  fim_de_semana = True
2  feriado = False
```

Outros tipos de dados

- **None** - usado para indicar a ausência de valor ou falta de um valor válido.

- **Dados binários:** frequentemente usados para manipular dados binários brutos, como dados de arquivos, protocolos de comunicação de rede, etc

  - ➔ **bytes**: É uma sequência imutável de bytes. Cada byte é representado por um valor inteiro entre 0 e 255.
  - ➔ **bytearray**: É semelhante ao tipo bytes, mas é mutável, o que significa que os elementos podem ser alterados após a criação.

Conjuntos de dados

- listas (list)
- tuplas (tuple)
- dicionários (dict)

Coleções de dados: **list**

As listas agrupam um conjunto de elementos variados, nelas podem estar contidos: ints, floats, strings, objects, outras listas etc.

Para definirmos uma lista utilizamos **colchetes para delimitar** a lista e **vírgulas para separar** os elementos:

```
alunos = ['Ana', 'Bruna', 'Davi']
notas = [8.5, 9, 7.5]
```

Coleções de dados: **tuple**

Semelhante às listas, as tuplas agrupam uma coleção de elementos de tipos variados. Porém, uma vez definida, as tuplas são **imutáveis**.

Utilizam **parênteses para delimitação** dos elementos e **vírgulas para separar os elementos**.

```
valores = ( 120 , 50 , 44 , 91)
```

Coleções de dados: **dict**

Os dicionários são utilizados para agrupamento de elementos através da estrutura de **chave** e **valor**.

Utilizam **chaves para delimitação** dos elementos, dois pontos para relacionar os valores e **vírgulas para separar os elementos.**

```
usuarios = { 'Alice': 12, 'Mário': 24, 'Leo': 32}
```

Conversão de tipos de dados

Conversão de tipos de dados

A conversão de tipos é uma operação comum e útil que nos permite transformar valores de um tipo de dado em outro.

Para esse tipo de operação, o tipo deve ser explicitamente definido.

```
cep = 25610041

cep = str(cep)
```

!! É preciso estar atento para a perda de dados no processo de conversão

Nomenclatura de variáveis

Boas práticas e regras para nomear variáveis:

Nomes descritivos

Nomes significativos e que descrevam o propósito ou conteúdo da variável

Letras minúsculas

Python é *case sensitive*. Por convenção, nomes de variáveis iniciam-se com letra minúscula

Evite caracteres especiais

Use apenas letras, números e sublinhados em nomes de variáveis

Evite palavras reservadas

Ex: ‘if’, ‘else’, ‘for’, ‘while’, ‘def’, ‘class’, ‘import’, ‘and’, ‘or’, etc

Estilo *snake_case*

Palavras separadas por sublinhados:
Ex: ‘saldo_conta’, ‘idade_usuario’

Convenções de nomenclatura

Recomendação: convenções de nomenclatura PEP 8

(https://peps.python.org/pep-0008/)